N90-23043

Algorithmic Considerations of Integrated Design for CSI on a Hypercube Architecture

Ü. Özgüner and F. Özgüner

Department of Electrical Engineering, The Ohio State University
2015 Neil Ave, Columbus, OH 43210

ABSTRACT

In this paper we present an approach to the integrated design problem for actively controlled large, flexible mechanical systems for which *Control Structure Interaction* (CSI) problems are of concern.

The two coupled design problems have been identified as the optimal *Structural Design* problem and the optimal *Controller Design* problem. These two problems can be addressed within a decision making loop that would consider each seperately, and then sequentially analyze the effects of one on the other. Embedded in such a loop would be the simulation and coordination tasks as part of the decision tools required in a total (software) package.

All of the above are compute–intensive tasks. In any such task, possible decompositions and gains due to the inherent parallelism have to be exploited. We claim that the problems under consideration, as applied to large flexible mechanical structures are particularly suited to be mapped onto multi–computer systems in a hypercube topology.

1. INTRODUCTION

Issues related to accomplishing integrated design for structural and control systems are of increasing concern in the context of Large Space Structures. Indeed a number of attempts have been made to come up with unified cost criteria and optimization approaches ([1], [2]). It is evident that one of the major hurdles in all such attempts is, and is going to be, computational. For truly large scale systems all four aspects, namely finite element modeling, control algorithms, over–all optimization and finally total closed–loop simulation singly or jointly create computational problems.

The use of distributed memory multiple processors connected in a hypercube topology has proven to be very useful in many large computation intensive tasks, especially with favorable parameter structures and algorithms which are compatible with the said topology. Indeed, the above four problems have been addressed individually (possibly in different contexts) for hypercube architectures. For example in [3], as a result of finite element discretization, linear equations of banded form are obtained and solved with an approach based on the *Conjugate Gradient* method. Experimental results on a 16–node Intel 386–based iPSC/2 hypercube have shown an almost linear speedup over a single processor implementation.

Some control design related work, specifically on solutions of quadratic regulator problems have also been reported in the literature [4] on hypercube based solution approaches. Yet these approaches have not been evaluated within the context of a total design package for Large Space Structures. In this paper we shall report on such an evaluation and introduce preliminary results for the development of a a CSI design package which assumes a large system with subsystems under decentralized control. The decentralized control design approach is based on the package DOLORES [5] which is being modified for the hypercube.

Distributed memory multiprocessors interconnected in a static topology such as a mesh, a toroid or a hypercube have been proposed as architectures particularly suitable for diverse application areas of scientific computing. However, in order to use these general purpose parallel computers in a specific application, existing algorithms need to be restructured for the architecture and new algorithms developed. In fact, conventional algorithms need to be reexamined, since the best algorithm for a sequential computer may not be the best for a parallel computer. Parallelization schemes for most applications on distributed memory multiprocessors are characterized by the mapping of a physical domain or its graph representation to processors with locality of communication. However, applications exhibit different characteristics in data dependencies and interprocessor communication patterns and volume. Finite element and matrix problems have very regular structures and the volume of communication and the amount of computation can be predicted.

Therefore, the basic model we shall consider is the banded matrix structure obtained from a finite-element model. We propose the retention of the nodal form of such models as we move from modeling to controller design and back. We intend to fix the number of nodes and nodal variables through the optimal design cycle. We will argue for the possible insertion of dummy nodes to retain as broad a *design space* as necessary for doing *parametric designs* where comparative evaluations of multiple criteria will becomeneeded.

The key reason for keeping the nodal form, possibly with excessive nodes, is doing the mapping to the processors only once. Thus, the required communication structure for the various problems will remain the same for different iterations of the same problem.

The configuration of a design package which includes the FEM stage is given in Figure 1.

2. THE FINITE ELEMENT MODEL

2.1 The Problems Considered

The idea behind the finite element method has always been to provide a formulation which can exploit the resources of digital computers. The resources provided by multiprocessors, particularly hypercube topologies, are especially useful for finite element based analysis and design problems where the banded structure of relevant matrices are exploited [6]. The important stages in such an analysis are:

- Obtaining the equations of motion of a structure by deriving the element equations and then *assembling* the equations for all elements.
- *Solving* the FEM equations, i.e. essentially solving a set of linear equations corresponding to static force–displacement type relations.

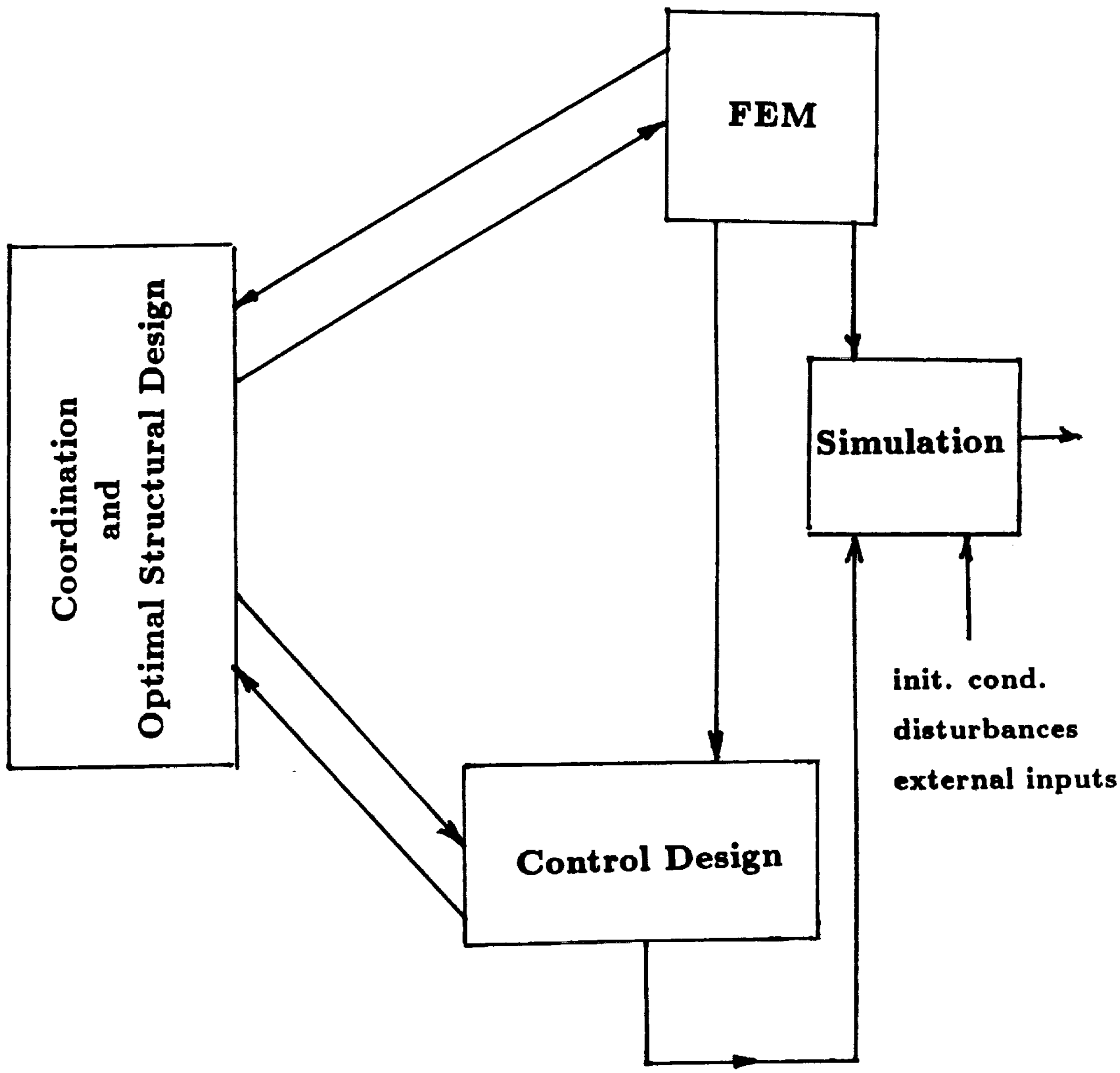


Figure 1: The configuration for a design package.

- *Modal analysis*, i.e. a dynamic analysis requiring solution for modes and mode–shapes.

Thus the problems of concern are solution of equations of the form,

$$\mathcal{K}q = \mathcal{B}F \tag{2.1}$$

for q and

$$\left(\mathcal{K} - \omega^2\mathcal{M}\right) q = 0 \tag{2.2}$$

for the set ω^2. The nxn $\mathcal{M}$ and $\mathcal{K}$ matrices are the mass and stiffness matrices respectively, and we assume that the consistent mass matrix approach has been utilized in generating $\mathcal{M}$. Thus the interdependancies implied by the elements of the two matrices (location of zeros, etc) are the same. The vector q is the nodal displacement vector, and F is the vector of applied forces to the structure and the matrix $\mathcal{B}$ denotes the influences on individual nodes.

2.2 Mapping to Processors and $\mathcal{K}$ Generation

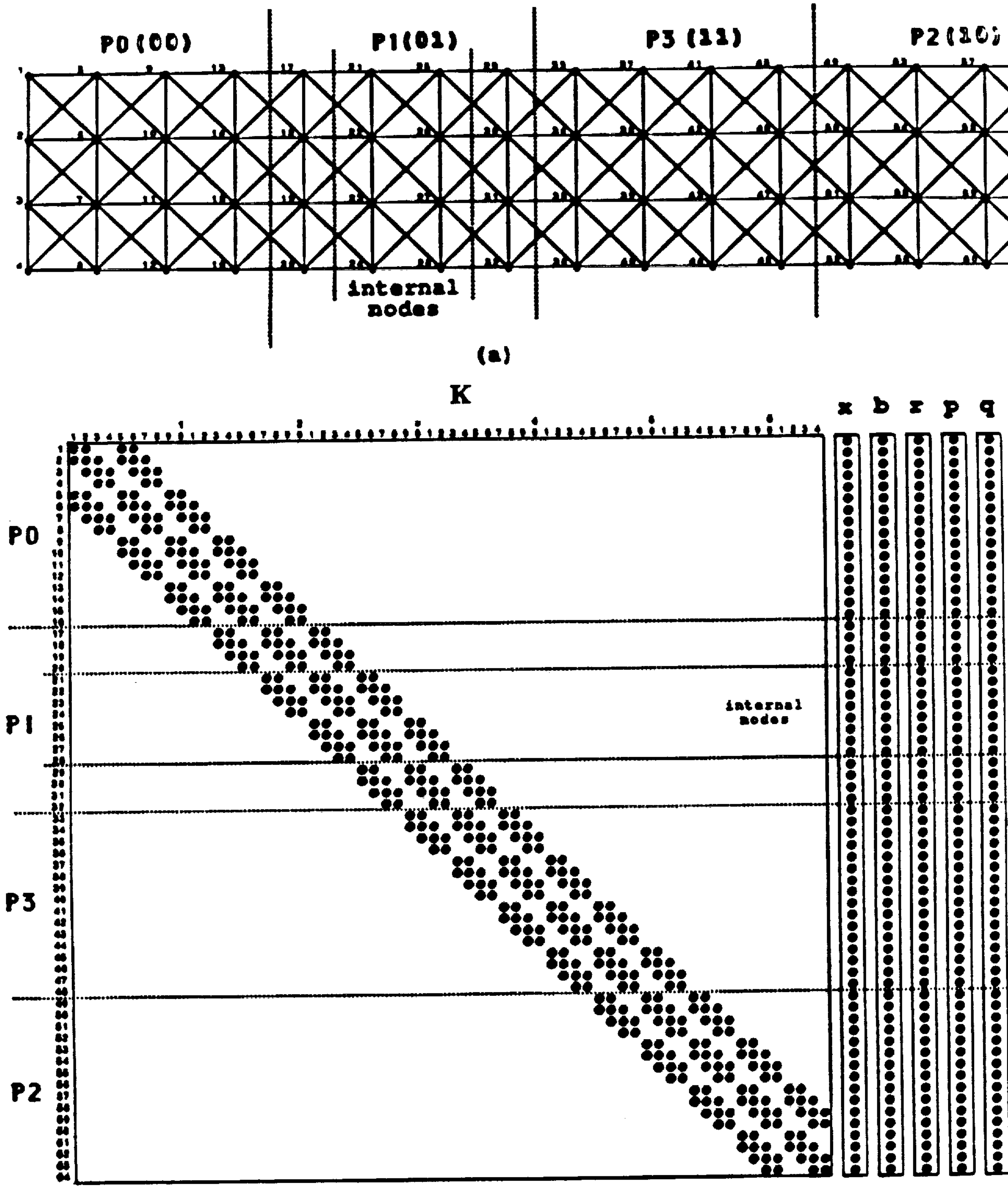
The key to having a good total design package is having properties in parameter sets that different portions of the package, addressing different problems, can jointly exploit. The basic property we want to exploit here is the banded form arising from the FEM. Thus the important first stage is obtaining that form and mapping it onto the processors [7]. This is coupled to the so–called $\mathcal{K}$ *Generation* operation. The elemental stiffness equations give relationships between each of the nodes associated with the element. The global $\mathcal{K}$ matrix is the superposition of the individual elemental K matrices. Therefore, the individual elemental K matrices may be computed independently and this portion of the generation of $\mathcal{K}$ can be done completely in parallel. It is important to note that the adjacency matrix associated with the finite element graph is identical in its zero–nonzero structure to the $\mathcal{K}$ matrix. This knowledge is utilized in a number of subsequent iterative solution schemes.

A row partitioning of the global $\mathcal{K}$ matrix corresponds to mapping a set of nodes onto each processor of a hypercube. The mapping determines which rows of $\mathcal{K}$ need be resident on a processor in order to perform the necessary matrix–vector product operations in any algorithm under consideration. The operations to be performed in parallel are then obtained from these rows.

The set of all elements which contain nodes in the mapped node set for a processor is *the associated element set* for that processor. If the nodes are contained entirely in the mapped set, the elements are *interior elements*. The difference between the associated and interior sets is called *the boundary element* set. This set provides information for the construction of rows mapped to at least two different processors and thus represents data that must be either communicated or calculated redundantly. [1]

Since the volume of data to be transmitted is proportional to the size of the boundary element set, it is important to perform the selection of this set carefully. An example of mapping is shown in Figure 2.

[1] The so–called Component Mode Synthesis techniques promise to provide a rich avenue of further research coupling both computation and control issues into basic FEM. The clear identification of the boundary element set provides insight into these relationships.



(b)

Figure 2. Strip mapping of (a) a finite element domain and (b) the corresponding **K** matrix onto a 2-d hypercube

2.3 Solving the FEM Equations

Methods for solving such equations on sequential computers can be grouped as: direct methods such as Gaussian elimination, LU decomposition and Cholesky factorization and iterative methods such as Gauss-Jacobi, Gauss-Seidel and Conjugate Gradient methods. Although extensive research has been done on parallelization of the solution of large sparse systems of linear equations, new architectural features such as the crossbar connection capability and massively parallel distributed memory architectures with fast communication offer the potential for new approaches to algorithm design.

Among the iterative methods, the CG algorithm [3], is increasingly being used in sparse matrix solvers, since it converges in at most n steps. The computational steps of the standard *CG* algorithm are given below:

Let $\mathbf{b} = \mathcal{B}F$. *Initially, choose* $\mathbf{x}_0$ *and compute* $\mathbf{r}_0 = \mathbf{p}_0 = \mathbf{b} - \mathcal{K}\mathbf{x}_0$. *Then, for* $k = 0, 1, 2, \ldots$

$$
\begin{array}{ll}
1. & \textit{form } \mathbf{q}_k = \mathcal{K}\mathbf{p}_k \\
2. & a.\ \textit{form } < \mathbf{p}_k, \mathbf{q}_k > \\
 & b.\ \alpha_k = \frac{<\mathbf{r}_k, \mathbf{r}_k>}{<\mathbf{p}_k, \mathbf{q}_k>} \\
3. & \mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{q}_k \\
4. & \mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k \\
5. & a.\ \textit{form } < \mathbf{r}_{k+1}, \mathbf{r}_{k+1} > \\
 & b.\ \beta_k = \frac{<\mathbf{r}_{k+1}, \mathbf{r}_{k+1}>}{<\mathbf{r}_k, \mathbf{r}_k>} \\
6. & \mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k
\end{array}
\qquad (2.3)
$$

Here, $\mathbf{r}_k$ is the residual error associated with the trial vector $\mathbf{x}_k$, *i.e.* $\mathbf{r}_k = \mathbf{b} - \mathcal{K}\mathbf{x}_k$ which must be null when $\mathbf{x}_k$ is coincident with $\mathbf{x}$ which is the solution vector and $\mathbf{p}_k$ is the direction vector at the k-th iteration. As seen from (1), the *CG* algorithm has three types of operations: matrix vector product $\mathcal{K}\mathbf{p}_k$, inner products $< \mathbf{r}_{k+1}, \mathbf{r}_{k+1} >$ and $< \mathbf{p}_k, \mathbf{q}_k >$ and the vector additions required in steps 3,4, and 6. To perform these operations concurrently, the rows of $\mathcal{K}$, and the corresponding elements of the vectors $\mathbf{b}$, $\mathbf{x}$, $\mathbf{r}$, $\mathbf{q}$ and $\mathbf{p}$ must be distributed among the processors by considering the communication features of the machine. Details of the implementation can be found in [3] where a speedup of over 15 is reported for a 16–node hypercube implementation.

2.4 Modal Analysis

Although much research on linear equation solutions for banded matrices has been done, results on eigenvalue calculations on hypercubes for banded matrices are very recent. We shall not dwell further on this issue except to point out that such an approach will have to be pursued to retain the advantages of the mapping described above.

3. INCLUSION OF ACTUATORS

3.1 The Proof–Mass Actuator Model

It is clear that the addition of an actuator with no bi–directional coupling with structure dynamics is simple and will only affect the mass matrix in the FEM. Thus the number of nodes in the FEM will not change. Some of them will have to be tagged in order to do a parametrized study of actuator location variation. On the other hand, actuators with dynamics coupled to the structure dynamics are somewhat more complex. In what follows, we consider such a case, specifically the model of a generic **proof–mass actuator**. (This follows the same lines as the analysis of a specific proof–mass actuator on a simple beam, presented in [8].)

Consider a proof–mass actuator attached to a flexible structure at a certain point. We shall simplify the problem somewhat by assuming that the point of attachment has been represented by a single node in the FEM. Let the nodal position variable (in the same direction as proof–mass movement) at this point be denoted by q_i. Let the proof–mass have mass m, friction constant d_m and spring constant k_m. [1] The equations of motion of the mass m are given by,

$$m\ddot{z} + d_m(\dot{z} - \dot{q}_i) + k_m(z - q_i) = f_e \tag{3.1}$$

where z denotes the position of the mass with respect to a fixed reference frame and f_e is the force generated by the windings and can be modeled as a linear DC motor as follows,

$$\begin{aligned} f_e &= k_e i & (3.2) \\ e &= Ri + k_b(\dot{z} - \dot{q}_i) \quad , & (3.3) \end{aligned}$$

where k_e, k_b are the motor constant and back e.m.f. constant respectively, R is the electrical resistence in the the motor circuit, and i and e denote the current and input voltage to the motor. Let us define the following constants:

$$\begin{aligned} d_e &= \frac{k_e k_b}{R} & (3.4) \\ b &= \frac{k_e}{R} & (3.5) \end{aligned}$$

Thus the total dynamics of the proof–mass actuator can be given as,

$$m\ddot{z} + (d_m + d_e)\dot{z} + k_m z - (d_m + d_e)\dot{q}_i - k_m q_i = be \tag{3.6}$$

The reaction force applied to the flexible structure is,

$$F = m\ddot{z} = f_e - d_m(\dot{z} - \dot{q}_i) - k_m(z - q_i) \tag{3.7}$$

[1] The last two could also have been added by electronic means, that is by wrapping a local feedback loop around the actuator. See [9] for such an example.

3.2 Embedding into the FEM

The actuator that has been analyzed in the present paper is particularly suitable for spacially decentralized control applications since it can be distributed over several locations of a given flexible structure for active vibration damping. Some attempts along these lines (using different devices) have already been reported in the literature [10,11]. To keep the notation simple, we shall continue treating the case with a single actuator. Consider now the dynamical equations representing the structure obtained from a FEM, with no actuator dynamics:

$$\mathcal{M}\ddot{q} + \mathcal{K}q = \mathcal{B}F \tag{3.8}$$

For our case $\mathcal{B}$ is a vector with all entries zero except the i'th which is 1. We define the expanded nodal position vector $\bar{q}$ as

$$\bar{q} = \begin{bmatrix} q_1 \\ \vdots \\ q_i \\ \vdots \\ q_n \\ z \end{bmatrix} \tag{3.9}$$

This is essentially equivalent to assigning nodal variables to the proof–mass actuator and considering it as a mass–spring system. The back e.m.f. and other effects have to be treated with some care. The final result will be an expanded FEM that can be written as,

$$\bar{\mathcal{M}}\ddot{\bar{q}} + \mathcal{D}\dot{\bar{q}} + \bar{\mathcal{K}}\bar{q} = \bar{\mathcal{B}}e \tag{3.10}$$

where the various matrices can be generated from the original (non–actuated) FEM and the actuator model. (An interesting and important observation here is that the structure will have damping with the input shorted, i.e. $e = 0$).

The point of all the above discussion is:

1. Actuator dynamics could have been introduced in the FEM stage.
2. If there are choices in actuator location, all "possible" actuators can be inserted by the introduction of dummy nodal variables.

4. THE OPTIMAL CONTROLLER DESIGN

4.1 A General Decentralized Structure

A number of results have appeared in the literature on using a quadratic regulator framework for controller design for flexible structures. We shall assume here that a decentralized information structure has been imposed. This means that there exists a correspondence between the inputs (actuators) and the output measurements (*sensors*). Let the system be modelled as,

$$\dot{x} = Ax + \sum_{i=1}^{\nu} B_i u_i \tag{4.1}$$

$$y_i = C_i x + D_i u_i \qquad ; \; i = 1, \cdots, \nu \tag{4.2}$$

where $x \in \Re^n$, $u_i \in \Re^{l_i}$, $y_i \in \Re^{m_i}$ and the matrices are real and of compatible dimension. Due to the decentralization constraint, only static feedback is allowed, the control is

$$u_i = K_i y_i \qquad ; \; i = 1, \cdots, \nu \tag{4.3}$$

or if dynamics are allowed in each feedback loop, the control is

$$\dot{z}_i = F_i z_i + G_i y_i \tag{4.4}$$

$$u_i = H_i z_i + N_i y_i \qquad ; \quad i = 1, \ldots, \nu. \tag{4.5}$$

where $z_i \in \Re^{q_i}$.

4.2 The Decentralized Quadratic Regulator

Consider a large scale system (4.1)-(4.2) with decentralized control (4.3) . The basic problem is to find an optimal static feedback gain so that the following cost function is minimized:

$$J = \int_0^\infty (x^T Q x + + \sum_{i=1}^{\nu} u_i^T R_i u_i) \, dt \tag{4.6}$$

and the following feedback structure constraint:

$$u_i = K_i \, y_i \qquad ; i = 1, \ldots, N. \tag{4.7}$$

It can be shown [12,13,5,14] that the necessary conditions for minimizing J given by (4.6) with the controller structure (4.7) imply the solution of the following system of nonlinear algebraic equations:

$$\begin{cases} A_c^T P + P A_c + \bar{Q} = 0 \\ A_c L + L A_c^T + X_0 = 0 \end{cases}$$

and

$$\nabla_{K_i} J = B_i^T P L C_i^T + R_i K_i C_i L C_i^T = 0$$

where

$$\begin{aligned} A_c &= A + \sum_{i=1}^{N} B_i K_i C_i \\ \bar{Q} &= Q + \sum_{i=1}^{N} C_i^T K_i^T R_i K_i C_i \\ X_0 &= x_0 \, x_0^T . \end{aligned}$$

4.3 The Lyapunov Equation Solution

It is obvious from the above that the solution of a single or coupled multiple Lyapunov equations,

$$AX + XA^T + Q = 0 \tag{4.8}$$

is key to many optimal controller design problems. Indeed the Lyapunov equation has been analyzed numerous times and many solution algorithms proposed. In general Schur decomposition based algorithms have been the accepted, reliable methods of solving small size problems, where A is dense and has no particular structure. Gardiner and Laub have addressed [4] the issue of large size and hypercube implementation, however the equations they have considered are still dense. Consideration of the sparse case is very recent, and general *heuristic* algorithms have been proposed in [15] with no convergence proof.

In considering linear equation solutions and modal analysis, we have discussed the case of banded matrices arising from the FEM. We have to, however, analyze clearly what the structure of the A matrix in the Lyapunov equation will be if it arises from a FEM.

Consider the *state equations* that one can obtain from the original model,

$$\mathcal{M}\ddot{q} + \mathcal{D}\dot{q} + \mathcal{K}q = \mathcal{B}F \tag{4.9}$$

If we go into *modal* form from the *nodal* form, with the unitary transformation,

$$z = \Phi^T q \tag{4.10}$$

to obtain,

$$\ddot{z} + \bar{D}\dot{z} + \bar{K}z = \bar{B}F \tag{4.11}$$

where the relevant matrices are,

$$\bar{D} = \Phi^T \mathcal{D} \Phi \tag{4.12}$$

$$\bar{H} = \Phi^T \mathcal{K} \Phi \tag{4.13}$$

$$\bar{B} = \Phi^T \mathcal{B} \tag{4.14}$$

we will obtain, in state space, the A matrix as,

$$A = \begin{bmatrix} 0 & I \\ -\tilde{K} & -\tilde{D} \end{bmatrix} \tag{4.15}$$

The required Lyapunov equations can then be solved in terms of the A matrix above. Note, however, that the problem is in modal space and the distribution to processors, accomplished during $\mathcal{K}$ generation is no longer valid. Besides, calculation of the nodal–modal transformations have to be accomplished.

Another approach which holds some promise is, remaining in nodal space and transforming the relevant control design equations. Thus, the equation of interest will no longer be the standard Lyapunov equation, but a transformed Lyapunov equation appropriate for systems modeled by second order differential equations. This provides the benefit of retaining the banded structure in the coefficient matrices without going through a modal transformation stage.

5. CONCLUSION

In this paper we considered a possible design package for large flexible mechanical structures that would address CSI problems. We advocate the exploitation of the banded structure that can be obtained from proper mappings in the FEM stage. We furthermore advocate the utilisation of distributed memory multiprocessors in a hypercube topology as particularly suitable for addressing the algorithmic problems.

Research in this area is somewhat new, but draws on aspects of previous work, in FEM, in matrix algorithms and in control design. The key issues are related to

- Being able to cast multiple problems into a single framework
- Retaining the same framework without doing unnecessary transformations
- Introducing dummy variables and parameters which can be used in parametric studies

Samples of subproblems and cases have been given to indicate the necessity of further work along these lines.

6. REFERENCES

[1] D. Miller, V. Venkayya, and V. Tischler, "Integration of structures and controls – some computational issues," in *Proc. of 24th CDC*, Ft. Lauderdale, FL, Dec. 1985.

[2] S.S.Rao, V.B. Venkayya and N.S. Khot, "Game theory approach for the integrated design of structures and controls," *AIAA Journal*, vol. 26, pp. 463–469, 1988.

[3] C. Aykanat, F. Özgüner, F. Ercal and P. Sadayappan, "Iterative algorithms for solution of large sparse systems of linear equations on hypercubes," *IEEE Trans. on Computers*, vol. 37, pp. 1554–68, 1988.

[4] J. Gardiner and A. Laub, "Hypercube implementation of some parallel algorithms in control," in *The Application of Advanced Computing Concepts and Techniques in Control Engineering*, NATO ASI, Denham and Laub eds. 1988.

[5] F. Khorrami, S. Tien, and Ümit Özgüner, "DOLORES: A software package for analysis and design of optimal decentralized control," in *Procedings of the 40th National Aerospace and Electronics Conference*, Dayton, OH, May 1988.

[6] L.M. Adams and R.G. Voigt, "Design, development and use of the finite element machine," Technical Report, NASA ICASE Report 172250, 1983.

[7] C. Aykanat, F. Özgüner, S. Martin and S.M. Doraivelu, "Parallelization of a finite element application program on a hypercube multiprocessor," in *Proc. 2nd Conf. on Hypercube Multiprocessors*, pp. 662–673, Knoxville, TN, Oct. 1986.

[8] J. Martin, Ü. Özgüner, S. Yurkovich, "An active vibration damper for flexible structures," in *Proc. of 17th Pitt. Conf. on Modeling and Sim.*, pp. 687–692, Pitt., PA, April 1986.

[9] E. Breitfeller and Ümit Özgüner, "Development of a control oriented model of a fixed-free beam with end mass," in *Proceedings of the American Control Conference*, Atlanta, GA, June 1988.

[10] U. Özgüner, S. Yurkovich, J. Martin, and F. Al–Abbass, "Decentralized control experiments on NASA's flexible grid," in *Proceedings of the 1986 American Control Conference*, pp. 1045–1051, Seattle WA, June 1986.

[11] N. H. McClamroch, "Vibration control of flexible structures using member dampers," in *1985 Decision and Control Conference*, pp. 936–939, Fort Lauderdale, Florida, December 11–13 1985.

[12] W. S. Levine, T. L. Johnson, and M. Athans, "Optimal limited state variable feedback controllers for linear systems," *IEEE Transactions on Automatic Control*, vol. AC–16, no. 6, pp. 785–792, December 1971.

[13] J. C. Geromel and J. Bernussou, "Optimal decentralized control of dynamic systems," *Automatica*, vol. 18, no. 5, pp. 545–557, 1982.

[14] F. Khorrami and U. Özgüner, "Frequency–shaped cost functionals for decentralized systems," in *Proc. 25th CDC*, Athens, Greece, Dec. 1988.

[15] A.S. Hodel and K.R. Poolla, "Heuristic approaches to the solution of very large sparse Lyapunov and algebraic Riccati equations," in *Proc. 27th Conf. on Decision and Control*, pp. 2217–2222, Austin, Texas, Dec. 1988.